## AMENDMENTS TO THE CLAIMS

Please amend the claims as indicated:

1. (Currently Amended) An apparatus for updating a code image, comprising:

a processor executing executable code stored on a main memory occupied by and used by an old code image and a temporary memory separate from the main memory~~storage device~~, the executable code comprising

a loader stored in the main memory and~~configured to~~ loading a new code image into the~~a~~ temporary memory ~~location separate from a memory space occupied by and used by an old code image~~;

a branch module stored in the main memory causing the processor to execute a bootstrap module within the new code image;

the bootstrap~~a logic~~ module ~~configured to~~ identifying incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image, and by accessing capability information for the old code image and capability information for the new code image and identifying a difference between the capability information~~;~~, and ~~a bootstrap module, within the new code image, configured to~~ reconcilinge the incompatibilities by changing an initialization order, ~~and~~ converting a format of a data structure of the old code image to a format compatible with a data structure of the new code image, and associating persistent data of the old code image with the new code image, such that the persistent data is available in response to execution of a run-time segment of the new code image; and

a copy module ~~configured to~~ copying the new code image into the main memory space occupied by the old code image.

2. (Currently Amended) The apparatus of claim 1, wherein the old code image is updated ~~substantially~~ concurrently with normal execution of transactions by the apparatus.

3. (Currently Amended) The apparatus of claim 1, the executable code further comprising an initialization module ~~configured to~~ initiatinge execution of thea run-time segment ~~of the new code image~~.

4. (Canceled)

5. (Canceled)

6. (Canceled)

7. (Canceled)

8. (Canceled)

9. (Currently Amended) The apparatus of claim 31, wherein identifying incompatibilities further comprises identifying a difference between the format of the data structure used by the old code image and the format compatible with the data structure used by

the new code image.

10. (Currently Amended) An apparatus for updating a code image, comprising:

a processor executing executable code stored on a main memory occupied by and used by an old code image and a temporary memory separate from the main memory~~storage device~~, the executable code comprising

a loader~~n update module configured to~~ loading a new code image into the~~a~~ temporary memory ~~location separate from a memory space occupied by and used by an old code image~~;

a branch module stored in the main memory causing the processor to execute a bootstrap module within the new code image;

the bootstrap~~a logic~~ module ~~configured to~~ identifying incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image, and by accessing capability information for the old code image and capability information for the new code image and identifying a difference between the capability information~~;~~, and ~~a bootstrap module within the new code image that executes subsequent to the update module, the bootstrap module configured to~~ reconcilinge the incompatibilities by changing an initialization order, ~~and~~ converting a data structure of the old code image to a format compatible with a data structure of the new code image prior to copying the new code image into the memory space occupied by the old code image, and associating persistent data of the old code

image with the new code image, such that the persistent data is available in response to execution of a run-time segment of the new code image.

11. (Currently Amended) The apparatus of claim 10, the executable code further comprising ~~wherein the bootstrap module is configured to reconcile incompatibilities based on the version information for the old code image and the new code image, and wherein~~ a copy module ~~is configured to~~ copying the new code image over the old code image in the main memory after the incompatibilities have been reconciled.

12. (Previously Presented) The apparatus of claim 10, wherein identifying incompatibilities further comprises identifying a difference between the format of the data structure used by the old code image and the format compatible with the data structure used by the new code image.

13. (Currently Amended) A system that overlays an old code image with a new code image with minimal interruption of operations being performed by execution of the old code image, the system comprising:

a main memory storing~~comprising~~ an old code image; ~~and~~

a temporary memory~~buffer~~ separate from the main memory and ~~configured to~~ storing~~e~~ a new code image;

a processor executing executable code~~instructions~~ of the old code image and the new code image, the executable code comprising~~to perform one or more operations, the processor configured to execute instructions of the old code image and the new code image~~;

a loader stored in the main memory and loading the new code image into the temporary memory;

a branch module stored in the main memory causing the processor to execute a bootstrap module within the new code image; and

the bootstrap module identifying incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image, and by accessing capability information for the old code image and capability information for the new code image and identifying a difference between the capability information, and reconciling the incompatibilities by changing an initialization order, converting a format of a data structure of the old code image to a format compatible with a data structure of the new code image, and associating persistent data of the old code image with the new code image, such that the persistent data is available in response to execution of a run-time segment of the new code image.

~~the memory further storing a pointer data structure configured to store an old code image pointer and a new code image pointer;~~

~~wherein, in response to an interrupt, the processor begins identifying incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image and executing bootstrap code within the new code image, the bootstrap code configured to reconcile incompatibilities between the old code image and the new code image by changing an initialization order and converting a data structure of the old code image to a format compatible with a data structure of the new code image.~~

14. (Currently Amended) The system of claim 13, the executable code further comprising a copy module stored within the new code image~~wherein the bootstrap code~~ overlaying~~s~~ the new code image in main memory with the old code image in response to reconciling~~ation of~~ the incompatibilities.

15. (Currently Amended) The system of claim 14, the loader loading the new code image into the temporary memory~~wherein~~ in response to an~~the~~ interrupt~~, the processor executes an update module of the old code image that loads the new code image into the buffer~~.

16. (Original) The system of claim 15, wherein the update module stores the old code image pointer and the new code image pointer in the data structure.

17. (Canceled)

18. (Currently Amended) The system of claim 33~~17~~, wherein the bootstrap module~~code~~ further reconciles the incompatibilities by updating modules that interface with the new code image.

19. (Canceled)

20. (Currently Amended) A method for updating a code image, comprising:

loading, by use of a processor, a new code image into a temporary memory location separate from a main memory space occupied by and used by an old code image;

executing a bootstrap module within the new code image;

identifying, using the bootstrap module executed by the processor, incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image, and by accessing capability information for the old code image and capability information for the new code image and identifying a difference between the capability information;

reconciling, using the bootstrap module executed by the processor, the incompatibilities by changing an initialization order and converting a data structure of the old code image to a format compatible with a data structure of the new code image using bootstrap code of the new code image, and associating persistent data of the old code image with the new code image, such that the persistent data is available in response to execution of a run-time segment of the new code image;

copying the new code image into the main memory space occupied by the old code image.

21. (Currently Amended) The method of claim 20, wherein the new code image is copied to the main memory~~old code image is updated substantially~~ concurrently with execution of regular computer operations.

22. (Currently Amended) The method of claim 20, further comprising initiating execution of the~~a~~ run-time segment of the new code image.

23. (Currently Amended) The method of claim 20, wherein the new code image is copied into the main memory ~~space~~ after the incompatibilities are reconciled.

24. (Canceled)

25. (Canceled)

26. (Canceled)

27. (Canceled)

28. (Currently Amended) The method of claim 34~~20~~, wherein identifying incompatibilities further comprises identifying a difference between the format of the data structure used by the old code image and the format compatible with the data structure used by the new code image.

29. (Currently Amended) An apparatus for updating a code image, the apparatus comprising:

a processor executing executable code stored on a main memory occupied by and used by an old code image and a temporary memory separate from the main memory~~storage device~~, the executable code comprising

means for loading a new code image into the~~a~~ temporary memory ~~location separate from a memory space occupied by and used by an old code image~~;

means for causing the processor to execute a bootstrap module within the new code image;

means within the bootstrap module for identifying incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image, and by accessing capability information for the old code image and capability information for the new code image and identifying a difference between the capability information;

means within the bootstrap module for reconciling the incompatibilities by changing an initialization order and converting a format of a data structure of the old code image to a format compatible with a data structure of the new code image using bootstrap code of the new code image, and associating persistent data of the old code image with the new code image, such that the persistent data is available in response to execution of a run-time segment of the new code image; and

means for copying the new code image into the memory space occupied by the old code image.

30. (Currently Amended) An article of manufacture comprising a program storage medium readable by a processor and embodying one or more instructions executable by a

processor to perform a method for updating a code image, the method comprising:

loading a new code image into a temporary memory location separate from a memory space occupied by and used by an old code image;

causing the processor to execute a bootstrap module within the new code image;

identifying, using the bootstrap module executed by the processor, incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image, and by accessing capability information for the old code image and capability information for the new code image and identifying a difference between the capability information;

reconciling, using the bootstrap module executed by the processor, the incompatibilities by changing an initialization order and converting a format of a data structure of the old code image to a format compatible with a data structure of the new code image using bootstrap code of the new code image, and associating persistent data of the old code image with the new code image, such that the persistent data is available in response to execution of a run-time segment of the new code image; and

copying the new code image into the main memory space occupied by the old code image.

31. (Currently Amended) The apparatus of claim 1, wherein the loader configures the temporary memory so that the executable code is executed directly from the temporary memory, the executable code further comprising an update module stored in the main memory maintaining an old code image pointer, a new code image pointer, capability fields storing the capability

information, an old code image version number, a new code image version number, the old code image pointer, the new code image pointer, the capability fields, the old code image version number, and the new code image version number used by the bootstrap module, wherein the bootstrap module follows the old code image pointer to locate an old code image header and a version field within the old code image header and follows the new code image pointer to locate a new code image header and a version field within the new code image header, the old code image header and the new code image header are organized according to the Microcode Reconstruct and Boot format, the bootstrap module reading the capability information from the old code image and the new code image and storing the capability information of the old code image and the new code image in the capability fields, the capability information comprising an indication that an EMULEX FLASH RAM is provided, the persistent data comprising login tables, ~~identifying incompatibilities between the old code image and the new code image further comprises accessing capability information for the old code image and capability information for the new code image and identifying a difference between the capability information~~ and wherein reconciling incompatibilities between the old code image and the new code image further comprises adjusting configuration settings and parameter lists.

32. (Currently Amended) The apparatus of claim 10, wherein the loader configures the temporary memory so that the executable code is executed directly from the temporary memory, the executable code further comprising an update module stored in the main memory maintaining an old code image pointer, a new code image pointer, capability fields storing the capability information, an old code image version number, a new code image version number, the old code image pointer, the new code image pointer, the capability fields, the old code image

version number, and the new code image version number used by the bootstrap module, wherein the bootstrap module follows the old code image pointer to locate an old code image header and a version field within the old code image header and follows the new code image pointer to locate a new code image header and a version field within the new code image header, the old code image header and the new code image header are organized according to the Microcode Reconstruct and Boot format, the bootstrap module reading the capability information from the old code image and the new code image and storing the capability information of the old code image and the new code image in the capability fields, the capability information comprising an indication that an EMULEX FLASH RAM is provided, the persistent data comprising login tables, ~~identifying incompatibilities between the old code image and the new code image further comprises accessing capability information for the old code image and capability information for the new code image and identifying a difference between the capability information~~ and wherein reconciling incompatibilities between the old code image and the new code image further comprises adjusting configuration settings and parameter lists and associating persistent data of the old code image with the new code image.

33. (Currently Amended) The system of claim 13, wherein the loader configures the temporary memory so that the executable code is executed directly from the temporary memory, the executable code further comprising an update module stored in the main memory maintaining an old code image pointer, a new code image pointer, capability fields storing the capability information, an old code image version number, a new code image version number, the old code image pointer, the new code image pointer, the capability fields, the old code image version number, and the new code image version number used by the bootstrap module, wherein the

bootstrap module follows the old code image pointer to locate an old code image header and a version field within the old code image header and follows the new code image pointer to locate a new code image header and a version field within the new code image header, the old code image header and the new code image header are organized according to the Microcode Reconstruct and Boot format, the bootstrap module reading the capability information from the old code image and the new code image and storing the capability information of the old code image and the new code image in the capability fields, the capability information comprising an indication that an EMULEX FLASH RAM is provided, the persistent data comprising login tables, ~~identifying incompatibilities between the old code image and the new code image further comprises accessing capability information for the old code image and capability information for the new code image and identifying a difference between the capability information~~ and wherein reconciling incompatibilities between the old code image and the new code image further comprises adjusting configuration settings and parameter lists.

34. (Currently Amended) The method of claim 20, the method further comprising configuring the temporary memory so that the executable code is executed directly from the temporary memory, maintaining an old code image pointer, a new code image pointer, capability fields storing the capability information, an old code image version number, a new code image version number, the old code image pointer, the new code image pointer, the capability fields, the old code image version number, and the new code image version number used by the bootstrap module, wherein the bootstrap module follows the old code image pointer to locate an old code image header and a version field within the old code image header and follows the new code image pointer to locate a new code image header and a version field within the new code image

header, the old code image header and the new code image header are organized according to the Microcode Reconstruct and Boot format, the bootstrap module reading the capability information from the old code image and the new code image and storing the capability information of the old code image and the new code image in the capability fields, the capability information comprising an indication that an EMULEX FLASH RAM is provided, the persistent data comprising login tables, wherein reconciling incompatibilities between the old code image and the new code image further comprises adjusting configuration settings and parameter lists.

35. (Currently Amended) The article of manufacture of claim 30, the method further comprising~~wherein~~ configuring the temporary memory so that the executable code is executed directly from the temporary memory, maintaining an old code image pointer, a new code image pointer, capability fields storing the capability information, an old code image version number, a new code image version number, the old code image pointer, the new code image pointer, the capability fields, the old code image version number, and the new code image version number used by the bootstrap module, wherein the bootstrap module follows the old code image pointer to locate an old code image header and a version field within the old code image header and follows the new code image pointer to locate a new code image header and a version field within the new code image header, the old code image header and the new code image header are organized according to the Microcode Reconstruct and Boot format, the bootstrap module reading the capability information from the old code image and the new code image and storing the capability information of the old code image and the new code image in the capability fields, the capability information comprising an indication that an EMULEX FLASH RAM is provided, the persistent data comprising login tables, ~~identifying incompatibilities between the old code~~

~~image and the new code image further comprises accessing capability information for the old code image and capability information for the new code image and identifying a difference between the capability information~~ and wherein reconciling incompatibilities between the old code image and the new code image further comprises adjusting configuration settings and parameter lists and associating persistent data of the old code image with the new code image.

36. (New) The apparatus of claim 29, wherein loading means configures the temporary memory so that the executable code is executed directly from the temporary memory, the executable code further comprising maintaining means stored in the main memory maintaining an old code image pointer, a new code image pointer, capability fields storing the capability information, an old code image version number, a new code image version number, the old code image pointer, the new code image pointer, the capability fields, the old code image version number, and the new code image version number used by the bootstrap module, wherein the bootstrap module follows the old code image pointer to locate an old code image header and a version field within the old code image header and follows the new code image pointer to locate a new code image header and a version field within the new code image header, the old code image header and the new code image header are organized according to the Microcode Reconstruct and Boot format, the bootstrap module reading the capability information from the old code image and the new code image and storing the capability information of the old code image and the new code image in the capability fields, the capability information comprising an indication that an EMULEX FLASH RAM is provided, the persistent data comprising login tables, and wherein reconciling incompatibilities between the old code image and the new code image further comprises adjusting configuration settings and parameter lists.